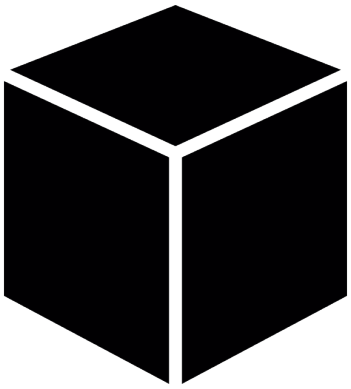
Vrije Universiteit Amsterdam

Accenture

How do you explain that?

An assessment of black box model explainers



Author
Marnix Maas

February 10, 2020

Abstract

Recent developments in machine learning have created methods to provide a human interpretable explanations for any predictive model. This research compares model-agnostic explainers Lime and Shap in three different simulated experiments. -It is suggested that Shap provides the best explanations for most models. However, the Lime explainer shows a marginally lower performance, while requiring less computation time. Secondly, a novel explainer evaluation metric is tested, and deemed inadequate in its current implementation.-

Revise

Might change layout of this page

Contents

1 Introduction

Decisions made by machine learning algorithms are often not interpretable by humans. Not even the most practised data scientists can expose exactly why their deep learning model decided the way it did. Still, the number of predictive models implemented grows each day. Most of which are so difficult to interpret, we tend to refer to them as black boxes.
In light of this problem, advances have been made to methods that provide an explanation for any model. Two of these so-called post hoc model-agnostic explainers are Lime and Shap. However complicated the predictive algorithm, these explainers intend to provide insight into all individual decisions. The model explainers simplify the model's decision into a human understandable explanation. For Lime and Shap the explanation takes the form of a sparse linear model. An intuitive example of such explanations is provided in Section 3.1.
The literature on model explainers is divided: arguments are put forward that the use of such methods is flawed or not desirable, while others encourage their use or are improving the explainers. I suggest that this division has occurred because the (non-)quality of explainers is difficult to measure. Making things worse, the more simplified the explanation, the more inclined it is to omit crucial information. If these challenges can be surmounted the improved accuracy of black box models can be utilized without sacrificing explainability.
Capabilities in model explanation methods are especially valuable for companies that market their machine learning expertise. Accenture advises other companies in opportunities with predictive algorithms and/or implements models to, e.g., improve efficiency. They can show how models provide value for other company, but will not be able to completely explain how predictions are formed when using black box models. Model explanation methods occupy this niche, as they allow Accenture to maximise value creation while maintaining the ability to explain any model used.

This research aims to evaluate and compare two recently proposed model explanation methods: Lime and Shap. A baseline for these explainers is an older method named 'Parzen windows'. Since no consensus for the evaluation of model explainers exists, the comparison is realised using two simulated user experiments previously proposed by Ribeiro et al. [1]. These experiments are constructed so that the working of the predictive algorithms is known. Hence, the explanations formed for each prediction can be evaluated. In addition, I propose several modifications in the implementation of

both experiments to improve evaluation of explainers. The experiments are performed on three real-world datasets and four sets of synthetically generated data. A second aim in this research is to analyse and assess a novel explainer evaluation metric called 'faithfulness' proposed by Arya et al. [5].

This study firstly introduces recent literature on the subject of post hoc model-agnostic explanation methods. Then, all implemented methods and their usage in the experiments are expanded upon. Figures and tables form an overview of the experiments' results and will thereafter be discussed. Lastly, the comparison of Lime and Shap is concluded and a discussion is established on implementations in this research.

2 Literature review

Recent developments and research on post hoc model-agnostic explainers is firstly discussed. Thereafter, a novel method to assess the quality of such explainers is covered. Lastly, tendencies against the use of explanation methods are considered.
It has been suggested that the terms explainability and interpretability should not be used as synonyms, as they may be different in nuance[1]. However, in this work these terms will both be used to describe the explanation formed by one of the discussed methods.

Post hoc model-agnostic explainers

In recent years, many advances in linear post hoc model-agnostic explainers have been made. The quest for such methods seems to come from the trade-off between model performance and explainability. Most recently best performing machine learning algorithms, while having improved accuracy over previous methods, are hard or impossible to interpret. For example deep learning and ensemble methods. As a result, a new field in model explainability has arisen. Most versatile are those methods that can be applied to any model - model-agnostic - and after the model has been trained - post hoc -.
To this end, Ribeiro et al. [1] have introduced Lime in 2016. Lime being a Local Interpretable Model-agnostic Explainer. In short, the method creates

[1] Interpretability would suggest a degree to which the model's output can be predicted. While explainability covers the degree of being able to explain to a human the inner mechanics of an algorithm.

a sparse linear explanation of an instance from the data that is locally faithful. The explanation consists of a maximum of K features, a user specified number, which are assigned an attribution to the prediction. The explanation is additive: all feature attributions (and a base value) sum up to approximately the model's prediction for that instance. The explanations are locally faithful in the sense that the explanation should be correct in the vicinity of the instance, but is not guaranteed to be globally faithful. In addition, the paper argues that Lime fulfills above mentioned properties using some simulated experiments.
An interesting addition to the explanation model is the notion of coverage. Their SP-Lime algorithm will find a set of the most representative explanations. That set of explanations should optimally describe the global behaviour of the algorithm.
In a work by Lundberg and Lee [2] all additive feature attribution methods, like Lime, are joined under a single definition. They unite Lime, DeepLIFT, layer-wise relevance propagation and Shapley value approximators (like in Strumbelj and Kononenko [11]). Then, it is shown that only Shapley values [10] can satisfy three desirable properties that enforce a unique solution for an additive feature attribution explanation. According to Shapley's theorem, that unique solution is optimal. Shortly, those properties describe zin herzien
local accuracy/faithfulness, no attribution if the feature is not included in the explanation and consistency of the feature attributions. Lastly, Lundberg and Lee propose a new method named Shap Kernel to approximate Shapley values with improved sample efficiency. It should be noted that Shap has brought the theoretical basis of Shapley values from game theory to additive feature attribution methods. However, it is assumed that features are independent and the explanation model is linear.
To counter these assumptions, an improved method was proposed by Aas et al. [3] in 2019. They suggest that the assumption of independency of features can produce faulty explanations. Thus, they propose an improvement of the Shap Kernel method to handle dependent features. Since the assumption of independency is only necessary in one step of calculating Shap values, they suggest relaxing that assumption. They found that for non-linear models the improved method outperforms Shap. A drawback of allowing dependencies is that explanations become harder to interpret. Multiple dependent features can only be properly interpreted as a group, rather than individually. Additionally, this extension to Shap increases computation time.
In addition to extending Shap for dependent features, an extensive overview of the Shap Kernel approximation method is given since the original paper [2] does not fully describe the implementation. An accurate description is

given of how the sample efficiency is improved over previous Shapley value approximations.

Aside from the creation of and improvements to model explainers, new methods have been proposed to evaluate them. A novel 'faithfulness' metric has been suggested by Arya et al. [5]. The method aims to evaluate the quality of an explanation using correlation. They have suggested a second evaluation metric named 'monotonicity', although that relies on similar evaluation of correlation. As of this date, no research has been done to evaluate their explainer faithfulness metric.

Disadvantages to post hoc explainers

The advances in explanation methods is compelling and shows potential for decreasing the trade-off between model interpretability and performance. However, some works have put forward that using post hoc explainers may not be beneficial.
In a recent paper from November 2019 by Slack et al. [7] a framework was created to fool Lime and Shap into producing faulty explanations. In short, the framework creates a clearly biased model and hides that bias by abusing the perturbations on which explainers rely. The explanations produced by the Lime and Shap seem unable to reproduce that bias. It should be noted that explainers are susceptible to "adversarial attacks" as Slack et al. suggest. Thus, if one has wrong intentions, Lime and Shap can be fooled into faulty explanations.
In some domains, for specific models, post hoc explainers have been shown to miss the most relevant feature of an instance. One such example is from Camburu et al. [4] for explaining a neural network for a natural language processing task. Their goal is to create (a framework for) an evaluation test for post hoc explanatory methods from the perspective of feature-selection. Their evaluation only considers the inclusion of features, unlike the faithfulness measure from Arya et al. While Camburu et al. conclude that post hoc explainers can miss the most relevant feature of a prediction, their framework was only tested on a single model for a single task. As they note in their conclusion, their framework could be applied to other tasks and areas.

3 Methods

This section will describe techniques used in the simulated user experiments. Firstly, an intuition of 'linear' model explanation methods is given. Model explainers Lime and Shap are discussed in detail, as they will be evaluated in the experiments. A modified version of a global explanation method called Parzen windows is used as baseline for Lime and Shap and will be shortly reviewed. In order to improve evaluation of explainers, two evaluation metrics are defined: faithfulness and NDCG. Lastly, the approach to synthetic data generation is considered.

3.1 Additive feature attribution methods

The explanation methods used in this paper fall into the category of 'linear explainers' or 'additive feature attribution explanation methods'. Linear comes from the fact that these explainers rely on sparse linear models for their explanations. The following sections will expand on the exact methods for Lime and Shap. In short, all explanations start at the average probability that the prediction belongs to that class, namely: 'base rate'. The explainer assigns an attribution to all features it includes, up to a provided maximum K features. Adding the attributions/impact of all features to the base rate results in (approximately) the algorithm's prediction. An example of additive feature attributions might be useful before going into detail.
Consider Figure 1 from Lundberg's repository[2]. In the example, the base rate starts at 0.1. The positive attributions displayed in green for the features Age, BP and BMI increase the probability by a total of 0.6. Whereas the negative attribution displayed in blue of feature Sex decreases the probability by 0.3. The sum of the base rate and all attributions comes out to be 0.4, which is (approximately) the algorithm's prediction.

3.2 Perturbation

Post hoc model-agnostic explainers rely on feature perturbation in order to provide explanations while treating the classifier as a black box. To measure the influence of a single feature on the prediction, it is taken out of the instance, i.e., the initial value . Most machine learning methods do not check allow a feature to have no value. Thus, the value is replaced in order to mask its impact. The value can be replaced by the most common/average value from the background dataset or a value of 0. A sensible choice for

[2]https://github.com/slundberg/shap

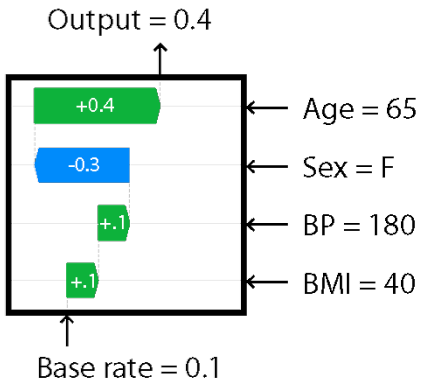


Figure 1: An example of additive feature attributions for a single prediction. Starting at the base rate (mean probability of that class) each feature contributes their attribution to end up at the output value. The positive attributions are shown in green. Negative attribution is shown in blue.

that replacement depends on the data. The process of masking a feature from an instance is called perturbation. An advantage is that perturbation is possible for any classification model.
Perturbation or the masking of features can also be done for multiple features of an instance. For Lime and Shap a perturbed sample of an instance x is denoted as z or z'.

3.3 Lime

The Lime (Local Interpretable Model-agnostic Explanations) method has been proposed by Ribeiro et al [1]. In short, Lime aims to provide a human interpretable explanation that is locally faithful to, and can be applied to any, machine learning model.
Firstly, interpretability, model-agnostic explainers and (local) faithfulness are characterized:

lime is perturbed differently. Normal dist and different for categorical features

- An explanation is interpretable if it can be easily understood by humans. For Lime specifically, it assumes that a sparse linear model is interpretable. Sparsity meaning that the linear explanation incorporates a small number of features that would allow for an easily readable explanation. E.g., a linear explanation with only 5 features should be quickly understandable for a human.

- The method is model-agnostic. The explanation method does not make any assumptions about the underlying predictive model. Therefore, it can be applied to any machine learning model.

- The explanations are locally faithful to the predictive model. According to Ribeiro et al. the explanation "must correspond to how the model behaves in the vicinity of the instance begin predicted" (2016: 3). Therefore, the explanation should be faithful to the model in a local manner, but it does not imply global faithfulness.

Formally, Lime has that g is an explanation with a limited number of features included. The complexity of g for linear explanations is measured by the number of included features and is denoted as $\Omega(g)$. G is the family of interpretable models (for example, a small decision tree would also apply). Let $f(x)$ be the probability that x belongs to a certain class according to model *f*. The proximity (vicinity) between instance x and z is $\pi_x(z)$, where z is a perturbed sample of instance x. Then Lime is defined by:

$$\xi(x) = \underset{g \in G}{argmin}\ L(f, g, \pi_x) + \Omega(g)$$

Lime finds the explanation $\xi(x)$ that is the most locally faithful, while still being interpretable. A formal definition of L is included in Appendix Section 8.1. The user can specify the complexity budget or maximum allowed number of features K. When the explanation becomes larger than K, the complexity constraint $\Omega(g)$ will be infinite. This enforces the explanation to be smaller than K. The explanation will look as described in Section 1. How the method works intuitively is shown in Figure 2.

3.4 Shap

A unified approach to model explanation methods named Shap, has been proposed by Lundberg and Lee [2]. In their work they aim to show that if a linear explanation is created for a model, Shap values are the most consistent and computationally viable.
Shapley values are a theorem from game theory, but they are used in the Shap explainer. Firstly, Shapley values are described. Thereafter, the use of Shapley values in the Shap explainer are considered.

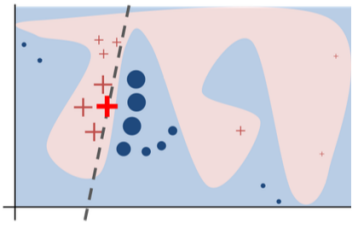


Figure 2: Source [1]. Intuition of Lime method: the linear explanation in dashed lines is constructed with weighted perturbed samples around the instance. The background colour indicates the decision boundary of the model. This figure aims to show that an explanation can be locally, but not globally, faithful to the decision boundary of the model.

Shapley values

Shapley values [10] are a method for distributing the total payout of several games over n persons, depending on their contribution to the payout for each game. According to the theorem, Shapley values are the optimal assignment of payout since they adhere to a set of desirable properties. The following section will cover these properties in full, as they are distinctly for additive feature attribution methods. a formal definition of the properties is included in Appendix Section 8.2. Young (1985) has shown that a solution that adheres to those desirable properties is unique. Thus, according to these theorems, Shapley values are the unique set of attribution for optimal distribution.
Molnar[3] (2019) accurately describes Shapley values: "A prediction can be explained by assuming that each feature value of the instance is a 'player' in a game where the prediction is the payout. Shapley values – a method from coalitional game theory – tells us how to fairly distribute the 'payout' among the features".

Unified definition - additive feature attribution methods

Shap Kernel is based on the Lime method, but with different choices for the weighting kernel π (distance metric) and regularization term Ω. For Lime, these parameters are chosen heuristically, whereas the Shap Kernel method defines these parameters according to the Shapley theorem. Formally, in

[3]https://christophm.github.io/interpretable-ml-book/shapley.html, chapter 5.9

Lundberg and Lee's theorem 2, the Shapley kernel is defined as:

$$\Omega(g) = 0,$$

$$\pi_{x'}(z') = \frac{M - 1}{(M \; choose \; |z'|)|z'|(M - |z'|))}$$

Where M is the number of features, the number of non-zero features in z' is $|z'|$ and x' is the set of all perturbations of the data. With this definition of the weight kernel and regularization, the estimated values should adhere to 3 desirable properties. Whereas the heuristic choice for Lime may result in a violation of local accuracy and consistency.

A formal definition of the desirable properties is given in Appendix Section 8.2. Intuitively, the properties can be explained as:

- Local Accuracy: The explanation exactly matches the predicted probability by the model for an individual. The mean explanation summed with all explained feature attributions is the same as the output of the model.
- Missingness: Features not included in the explanation do not have any attribution to the explanation.
- Consistency: Feature attribution should not decrease if the features' input is kept the same or increased, while the other inputs are unchanged.

A unique additive feature explanation model follows from these properties:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!}\,[f_x(z') - f_x(z' \setminus i)]$$

Where i indicates a specific feature.

Shap Kernel

The complexity of calculating Shapley values depends on the number of features. The exact calculation of Shapley values becomes computationally intractable for a large number of features. Since the exact calculation is computationally expensive, Lundberg and Lee have proposed Kernel Shap to approximate them. Under the assumption that features are independent and the model is linear, Shap values can be calculated with higher sample

efficiency than previous Shapley equations. It should be noted that feature independence is an assumption when using perturbations. The assumption is not specific to Shap values. The improved sample efficiency is described in detail in a different paper in Section 2.3 by Aas et al. [3]. They elaborate that the calculation of Shap values is made more efficient by performing part of the matrix calculations once for several explanations at a time, instead of once for each explanations. In addition to the model-agnostic Shap Kernel method, Lundberg and Lee have introduced several model specific methods for calculating Shap values even more efficiently. These include Deep Shap for deep learning models and Tree Shap for tree based models. However, for this research only true model-agnostic methods are considered.

In summary, Shap introduces a faster method to approximate Shapley values. According to the theory, only Shapley values adhere to a triplet of desirable properties for an explainer. Like the Lime method, Shap will produce an explanation with feature attribution for up to K features.

3.5 Parzen

The Parzen-windows technique is an approach to estimate the probability density function of a specific point without knowing the underlying distribution. A region around the point is used to estimate the value of probability density. That region is where the name Parzen-windows comes from. Baehrens et al. [9] describe how the Parzen-window can be used to explain individual classification decisions.
They define the Bayes classifier:

$$g^*(x) = arg \min_{c \in \{1,\dots,C\}} P(Y \neq c | X = x)$$

where C is the number of classes in the classification problem and $P(X, Y)$ is some unknown joint distribution. Then, the explanation vector of a data point x_0 is the derivative to x at $x = x_0$. Formally noted as:

$$\zeta(x_0) := \frac{\partial}{\partial x} \, P(Y \neq g^*(x) | X = x)\Big|_{x=x_0}$$

With $\zeta(x_0)$ a d-dimensional vector, with the same length as x_0. The explanation is formed by the largest (absolute) feature attributions in the vector $\zeta(x_0)$, up to K features. Note that Lime defined the explanation vector as $\xi(x)$.

3.6 Faithfulness metric

The quality of an explainer depends on the interpretability offered to the user as well as the faithfulness to the model. These notions may have an opposing effect. A simpler explanation may be less faithful to the model, as it cannot capture its full extent.
Current post hoc explanation methods use a (sparse) linear explanation. They should offer the same level of interpretability as long as they have the same number of features. Thus, they could easily be compared based on some notion of faithfulness to the model.
Currently, there is no standard metric to measure that faithfulness. However, Arya et al. [5] have introduced a conveniently named faithfulness metric called: 'faithfulness'. This metric aims to quantify the quality of explainers, rather than human evaluation being the golden standard. In this paper, the aim is to determine the quality of this metric.
The faithfulness metric expresses the quality of an explainer as correlation between model predictions and feature importance. In order of feature importance, the feature of an instance is replaced by the background value and the model's prediction probability is recorded. That process is repeated for all features for which a feature importance exists. The faithfulness metric ϕ is then defined as the negative Pearson correlation ρ between the vector of feature importances Θ and the vector with the model's prediction probabilities $\mathbf{p}$:

$$\phi = -\rho(\Theta, \mathbf{p})$$

The higher ϕ the better the quality of an explainer (beware not to confuse this ϕ with feature attributions from previous definitions). Intuitively, the model's prediction probability should decreas when a feature with positive attribution is removed. Thus, the faithfulness metric could show to what degree that intuition is followed. Intuitively, the method scores explainers for attribution values that have the same impact as the model when excluding/perturbing the feature.
The method has a drawback, as this metric uses correlation it is not defined for small explanations. It is not possible to calculate correlation when the length of Θ and $\mathbf{p}$ is 1, as correlation is not defined for a point. And if they consist of two probabilities and feature importances, the correlation is always either 1 or -1. The metric would always assign either the best or worst score to the explanation. Thus, this is not expected behaviour for such a metric. Nonetheless, explanations would often consist of more features. For these explanations the faithfulness metric could still provide insight, by scoring the intuition as described above.

3.7 Ndcg

Normalized discounted cumulative gain is a measure of ranking quality. The metric is mostly applied in information retrieval. Search algorithms are scored by their ability to retrieve the most relevant documents in order. I propose to use this metric in the evaluation of model explainers. This popular technique not only assesses the explainers by the features it retrieves, but also the ranking of features. Formally, ndcg is defined as:

$$ndcg_p = \frac{dcg_p}{idcg_p}$$

where

$$cdg_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)}$$

and

$$idcg_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

rel_i is and individual feature i and $|REL|$ is the list of features ordered by importance. Intuitively, the metric will assign a score between 0 and 1, comparing the ranking of features by the explanation to the true rank of features according to the model. A value of 1 indicates a perfect ordering.

3.8 Synthetic data generation

For the two experiments synthetic data is generated. With synthetic data the amount of features, dependencies, noise and other factors can be controlled. Then, the robustness of model explainers can be evaluated given the modifications to the data.
Data is generated using the make_classification function from sklearn[4]. The package provides a method to generate a dataset with user specified modifications. Firstly, the user defines the number of informative features. Then, a number of redundant features can be specified. These redundant features are a random linear combinations of the informative features from the dataset. In addition, noise is created by replacing the target variable with a randomly selected target output. The following section will elaborate on other parameters used in the data generation process.

[4]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

4 Experimental setup

There is no consensus for the evaluation of model explainers. Hence, the comparison is done via simulated user experiments. These experiments are constructed so that the working of the predictive algorithm is known and can thus be used to quantitatively assess the explainers. The experiments are based on those presented in a paper by Ribeiro et al. [1]. Their code is provided in an online repository[5]. This is the basis for the experiments in the current paper and they will accordingly be named (Lime) Experiment 5.2 and 5.3. Minor adjustments have been made to run this experiment in Python 3.7 rather than 2.7.

In order to improve the measurement of explainer quality, several adjustments for the implementation of both experiments are proposed. Those improvements include the two explainer quality metrics: faithfulness and NDCG. Additionally, both revised experiments are applied to synthetically generated data. Since that data can be manipulated to test the explainers in their handling of redundancy and noise. Following subsections will describe all experiments in detail. Code for the current study is available at: https://github.com/marnixm/lime_experiments. URL nog wijzigen

While Aas et al. [3] suggest that using Shap with an extension for dependent features may be beneficial, its explanations are not as simple as Lime and Shap provide. Its explanations can only be properly interpreted as clusters of dependent features. For the current study only true additive feature attribution methods are considered. Thus, Shap with the extension for dependent features is not included in the experiments.

The original Lime experiments include a random and greedy explainer. However, they only provide inclusion of features, but not feature attribution. Since feature attribution is crucial to explainers, the greedy and random explainer are insufficient as baseline. Hence, they too have been excluded from the experiments.

As mentioned in the faithfulness section, the metric is not defined for explanations of two or fewer features. Instances to which this applies have been omitted from the experiment's results.

All explainers are provided a maximum budget of $K = 10$ features for their explanations. The Parzen explainer may find explanations of larger size, unlike Lime and Shap. Therefore, the Parzen explanations will consist of only the K most important features (as was the choice in the original Lime

[5] https://github.com/marcotcr/lime-experiments

experiments).

4.1 Real-world data

The data consists of product reviews from Amazon.com. A review is marked as either positive or negative, thus a binary outcome. The data has been used for several studies, initially by Blitzer et al. [6]. The features in this dataset are the words used in the reviews for each domain. For the current study, the datasets on DVDs, books and kitchen products have been used.
All three datasets have approximately 20.000 features and 2000 rows of data. The data is split into a train and test set of respectively 1600 and 400 rows.

4.2 Lime experiment 5.2 Are explanations faithful to the model

In this experiment, machine learning methods are used that are interpretable by themselves. Namely sparse logistic regression and decision trees. However, these models are only allowed to use a maximum of $K = 10$ features for each row in the test set. Thus, for all instances, a golden set of features is known. In this experiment, the writers aimed to show that their explainer can find the features used by the model.
A comparison is made of the golden set of features for each instance to the explanations. Thus, scores for recall of golden features from the model can be calculated. Precision would also be an interesting metric to consider. However, the models are asked to provide an explanation of 10 features. If the model itself would use less than 10 features, the explainer could never reach a precision of 1.
The sparse logistic regression model uses L1 penalty, where the penalty parameter is increased until a maximum of 10 features for each row is used. Similarly, the decision tree model is only allowed to use a maximum of 10 features.
It should be noted that while Section 5.1 from the Lime paper [1] describes the use of L2 regularization for the linear regression in their experiments, it is actually L1 regularization that was implemented in their repository. Both Shap and Lime are provided with a budget of 15.000 samples.
Lastly, beware that the title of this experiment is not to be confused with the faithfulness metric. The title actually advertises that the explainer should use the same features as the models themselves: recall.

4.3 Lime experiment 5.2 improvements

Number of golden features for each instance

The sparse linear regression model from experiment 5.2 is provided with an increasingly high penalty until a maximum of 10 features are used for all instances. More precisely, the features used by model (Θ) and the features provided by the explainer (ξ) are for **all** instances: $|\Theta \cap \xi| < 10$. In practise, the sparse linear regression is provided with such a high penalty, that this intersection has an average of 2-3 features. Thus, the explainer is allowed to provide 10 features, whereas on average 2-3 features are used per instance. As a result, high recall numbers of $> 90\%$ are reported.

sub headers

Instead, it is suggested to find models with an average (rather than maximum) of 10 used features for each instance. For the calculation of recall and faithfulness, only the 10 most important features are considered. In the original experiment, the explainer were allowed a larger budget to retrieve all features ($|\xi| > |\Theta|$). Now, the explainers will have to retrieve an amount of features that is on average equal to their budget ($|\xi| \approx |\Theta| \approx K$). By increasing the amount of golden features, the test should be more challenging for the explainers.

However, for the decision tree model it is not reasonably possible to increase the amount of golden features. Initially, the model uses 200+ distinct features in the complete tree. However, an instance walks a specific path among the branches of the tree in order to reach a prediction. This path will in all likelihood not reach every feature, thus the golden set of features is smaller than the 200+ from the complete tree. For the real-world datasets used, the average amount of golden features is actually only 1.4 to 1.5. It is possible to increase the size of the tree. However, the tree would be specifically trained so that it uses an increased amount of features. Where a decision tree would normally be trained to optimize each split. In other words, the larger decision tree would not represent a model build in any real world situation. Therefore, the tree model is not modified. Only the logistic regression model will increase the amount of golden features.

Table 1 shows an overview of the average amount of golden features for the improved experiment.

	Books	DVD's	Kitchen
Logistic regression	6	5	7
Decision tree	1.5	1.4	1.4

verify numbers

Table 1: Average number of golden features for each instance

Ranking of explained features

Furthermore, only the inclusion of features has been considered so far. Even thought all these explanation methods also provide feature attribution: a degree of impact to the prediction. Whereas the faithfulness measure tries to evaluate each attribution, I would argue that the ordering of variables should provide a better measurement than recall alone. An explainer is even better if it can find the golden set of features, but is also able to rank them in order of importance. Hence, it is proposed to take ranking of variables into account. To this end, ndcg is used to calculate the quality of explanations. For the logistic regression model features are ordered by their (absolute) coefficient value. The order of features for the decision tree model is determined by their variable importance. Secondly, to calculate ndcg both the explainer and instance have to be of the same length. The ndcg is calculated for the x most important features of these instances. Where x is the minimum length of the explainer or instance. However, if the explanation uses less features than the instance it is not penalized. To counter this flaw, the ndcg score is multiplied with the recall for that instance. If the explanation is too short, it will be penalized by the recall score. When the instance uses less features than the explanation, we consider only the amount of features from the instance. After all, the explainer should not have returned additional features for its explanation.

To summarise, it is suggested to consider the ndcg score for evaluation of explanations, since the ranking of variables is taken into account. To adjust for explanations of unequal length, we penalize using recall.

Synthetic data

Data is generated so all intricacies in the data can be modified. The datasets will be set-up similarly to the real-world datasets in this research, but with varying levels of noise and dependency. All datasets will have 10 informative features, the same amount of features the explainers are allowed to provide. From these 10 features, either 0 or 15 redundant features are created. Extra random (useless) features are added so that the total number of features is 50. The noise parameter fluctuates between 0.05 or 0.3. Thus, four datasets are generated with low and/or high amount of noise and redundancy. All datasets will have 2000 rows and are similarly split into a train and test set of respectively 1600 and 400 features.

4.4 Lime experiment 5.3 Should I trust this prediction

Trustworthiness of the explainers is assessed by considering the impact of features on the model and the explanation. In this experiment it is assumed that the user can identify a number of features that are not trustworthy and that the user would not want to include in the model. For each instance in the dataset, 25% of the features are sampled to be untrustworthy. We then compare how the predictions of the model and explainer change by removing the effect of the untrustworthy features. In other words, the user 'discounts' the effect of untrustworthy features.
Discounting for the model is done by replacing untrustworthy feature values with the background value. The adjusted instance is passed through the classifier for a new predicted probability. In case of the explainer, the initial prediction is formed by the (local) mean prediction added to all explained features attributions. Discounting for the explainer is the initial prediction subtracted by the explained model impact of each feature that is untrustworthy in that instance.
For both the model and explainer, the experiment will compare the initial prediction with the prediction after discounting untrustworthy features. If the untrustworthy features change the prediction of the model, the explainer should show the same behaviour. It is pointed out that the classification problem is binary. Thus a change of prediction is defined by going from negative to positive classification, or vice versa.
Considering the change of prediction results in a vector of trusted and mistrusted instances, precision and recall can be calculated from these vectors. In the original paper, the F1 score was reported for two datasets.
Trustworthiness can be tested for any classification model. Five models are considered in this experiment: sparse logistic regression (LR), nearest neighbours (NN), random forest (RF), support vector machine (SVM) and decision tree (Tree). Model parameters are as in the Lime paper. Only the solver for the logistic regression was changed to 'lbfgs', since the old solver was no longer supported. For this experiment the faithfulness metric is not implemented. This experiment chooses an arbitrary amount of untrustworthy features and discounts them from the explanation. By design, the explanation before and after discounting should not differ in quality. Hence, the performance in this experiment cannot be measured with the faithfulness metric.

4.5 Lime experiment 5.3 improvements

In the original paper, it was decided to compare whether both the model and explainer changed predictions due to the removal of untrustworthy features. However, the model and explainer should in the first place have the same prediction. While this is always the case for Shap values (due to local accuracy guarentees), Lime and Parzen explanations may initially be wrong[6]. This had not been taken into account in Robeiro et al. [1]. To adjust for this effect, a new 'local accuracy' score is introduced. For convenience, the score will be referred to as accuracy. The accuracy of the explainer is decreased if the prediction, before removal of untrustworthy features, is not the same as the model. The improved experiment proposes a penalization of the f1 score if the local accuracy property is violated. Experiment 5.3 will present this adjusted f1 score.
Parameters. Generated 50 features, 10 informative, 0/15 redundant, 0.05/0.3 noise

4.6 Experiments with generated data

Synthetic data is generated so that modifications to the data can be specified. The controlled modifications may show weaknesses of explainers. The improved experiments 5.2 and 5.3 are repeated using the synthetic data. Several adjustments were made to allow the use of synthetic data.
Firstly, to provide Lime explanations the lime tabular package is implemented. The standard lime kernel has been used. In addition, the explainer has to be provided with a background dataset for perturbations. Shap explanations now too require a background dataset. Due to time limitations the background dataset is summarised using K-means clustering (as Lundberg suggests in his repository[7]). For this experiment, the data is summarised in 10 clusters. The Parzen windows explainer required only two parameters. These parameters are equal to those used for the Books dataset. Lastly, the faithfulness metric still requires a background dataset for perturbations. For this the average value of each features is used.
In accordance to the improvements from experiment 5.2, the average number of golden features for each dataset is reported in table 2.

[6]Note that the model has a binary outcome, predictions side with positive or negative at the threshold of 0.5.

[7]https://slundberg.github.io/shap/notebooks/Iris%20classification%20with%20scikit-learn.html

	Gen1	Gen2	Gen3	Gen4
Logistic regression	10	10	10	10
Decision tree	10	10	11	11

check numbers

Table 2: Average number of golden features for each instance

5 Results

This section will present the results of all simulated user experiments described in the previous section. These experiments aim to measure the performance of explanation methods.
Firstly, the original experiment 5.2 as in the Lime paper is presented. Secondly, the results from the improved experiment is shown. Thirdly, we repeat the improved experiment with synthetic data. Then, the faithfulness metric is tested. In similar order, the outcome of experiment 5.3 is presented afterwards.

Experiment 5.2

In this experiment small interpretable models are trained such that a golden standard of features is known. It should be noted that the average number of true features, is only between 2-3 . Whereas the explainers are given a budget of 10 features for their explanations. Explanations are generated for each instance in the test dataset. The average recall is shown in Figure 3. Even though the explanation methods are model-agnostic, the figure shows

refer to table

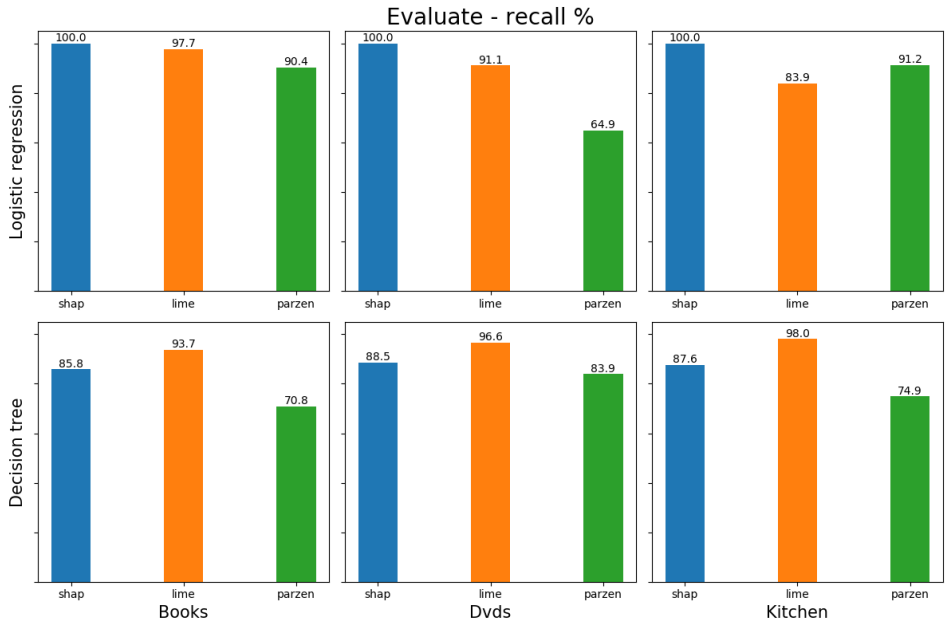


Figure 3: Experiment 5.2 recall %

varying results.
Looking at the logistic regression model, it can be seen that Shap is able to retrieve all features for these datasets. Lime produces the second highest recall for two of the three datasets. Only for the Kitchen dataset, is Lime outperformed by Parzen[8]. In contrast, for the decision tree model Lime has dominant recall over the other explainers. Shap retrieves the second highest amount of features, still more than the baseline for all datasets.
According to these results, an argument is put forward that Shap's assumption of independent features is punished by models that allow that dependency, decision trees in this case. As is represented by a decreased recall score for Shap with the decision tree model. This speculation would agree with the given results.

Experiment 5.2 improved

For this experiment, the amount of true features used for each instance has been increased. Such that the average (rather than maximum) amount of golden features is approximately 10. For some instances, the logistic regression uses over 50+ features in the golden standard, but these instances only consider the 10 most important features. As a result, the presented recall scores are lower in comparison to the original experiment. The number of golden features for the decision tree has not changed.
The recall scores for this improved experiment are shown in figure 4. The figure shows that for the logistic regression model all recall scores are lower in comparison to the original experiment. This would be expected since the explainers now have to retrieve an amount of features (approximately) equal to their budget. The compared performance of the explainers has not changed. Though, it should be noted that Shap no longer has a perfect recall score.
Ndcg scores for the explainers are presented in figure 5. Not only is feature inclusion measured, but the ranking of features is now taken into consideration as well. For the logistic regression model the explainers perform worse when compared to just their recall scores. However, the impact on Shap's scores is smaller than for the other explainers. The ndcg scores for the decision tree model seem to have decreased more than with logistic regression. The figure shows that Lime still outperforms Shap on decision trees, whereas they both perform better than the baseline.

check 50+

mention decision tree not changed?

rewrite this paragraph

[8]Interestingly, that particular dataset was excluded from the results section of the original paper.

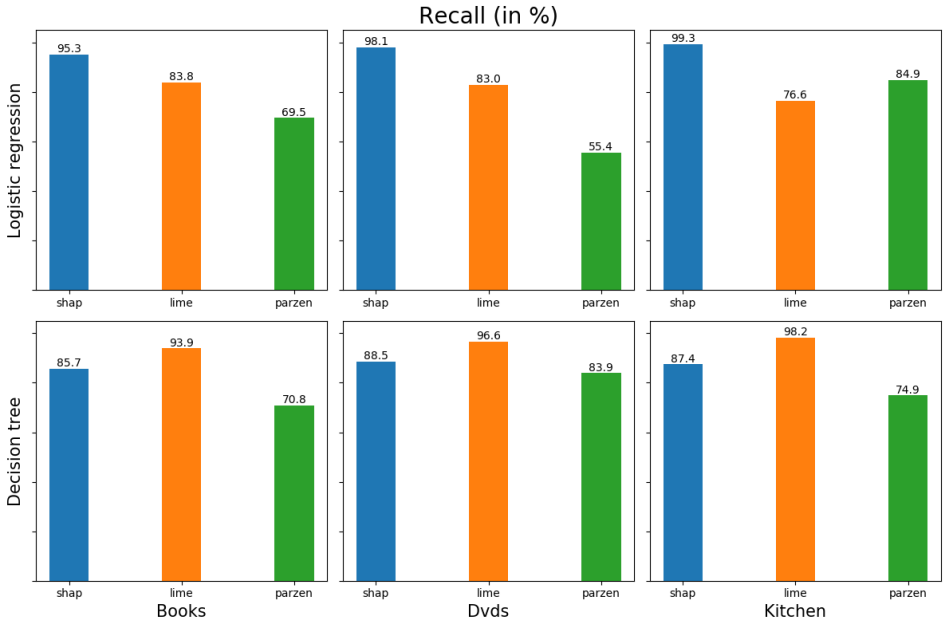


Figure 4: Experiment 5.2 recall metric %

I suggest that the ndcg scores present a better image of explainer quality, since it allows the measurement of ranking of features. Furthermore, when ndcg is corrected with recall, the measurement is also able to deal with explanations that involve less features than the provided budget K.

Experiment 5.2 Generated data

Synthetic data is generated so that the underlying dependencies and noise can be controlled. This data allows for a better assessment of the explainers as the impurities of a real-world dataset are not present.
Recall scores are shown in figure 6. It shows comparable results for Lime and Shap for the logistic regression model. Interestingly, for the decision tree model Lime is only able to outperform Shap with the least modified data. Once dependent features and more noise is introduced, Shap is able to retrieve a larger amount of features.
Figure 7 presents the ndcg scores for generated data. The best results fluctuate between Lime and Shap for all datasets and models. Although, Lime and Shap score comparably for the decision tree model. Since Lime scored worse in terms of recall, the explainer must have better ranking of features

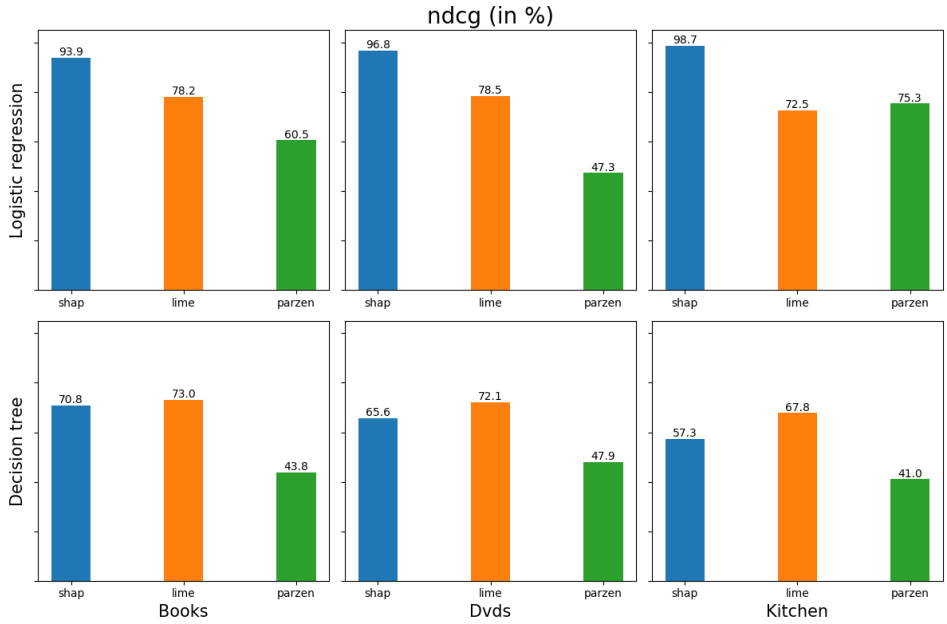


Figure 5: Experiment 5.2 ndcg metric %

for the generated data, in order to regain competition with Shap. rewrite

Faithfulness metric

The average faithfulness scores for this experiment (original, improved, generated data) are respectively shown in Figures 8, 9, 10. Note that the higher the faithfulness score ϕ, the better the explainer should be. The figure demonstrates that Shap has dominantly higher faithfulness scores than the other explainers on all datasets. In second place comes Lime with scores fluctuating from approximately -2% to 8%. Lastly, the baseline often produces the lowest scores. check numbers

However, upon further investigation the faithfulness measure seems impacted by small feature attributions. Since the explainers are asked to find an explanation of size $K = 10$, they will assign some features a small attribution. Even if, for that instance, less than 10 features are used. Remind that the experiment resulted in an average of 2-3 features per instance.

Scores where the faithfulness is nearly one, seem to come from above described explanations. The model's prediction does not change, because the features are not included. The explainer however, has assigned a tiny at-

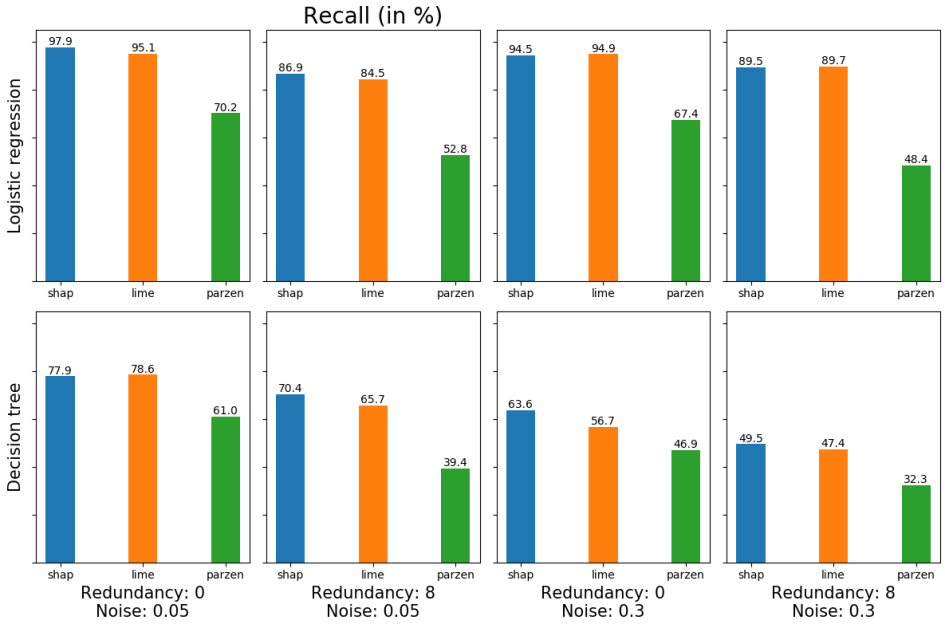


Figure 6: Experiment 5.2 recall metric %

tribution. A consequence is that these explanations are assigned scores of approximately 1 and -1. Especially for Lime, the faithfulness scores fluctuate between these extremes. As a result, the average faithfulness is close to 0.

It is argued that this behaviour of the faithfulness metric is not desirable, having in mind that the explainers are forced to a K-sized explanation.

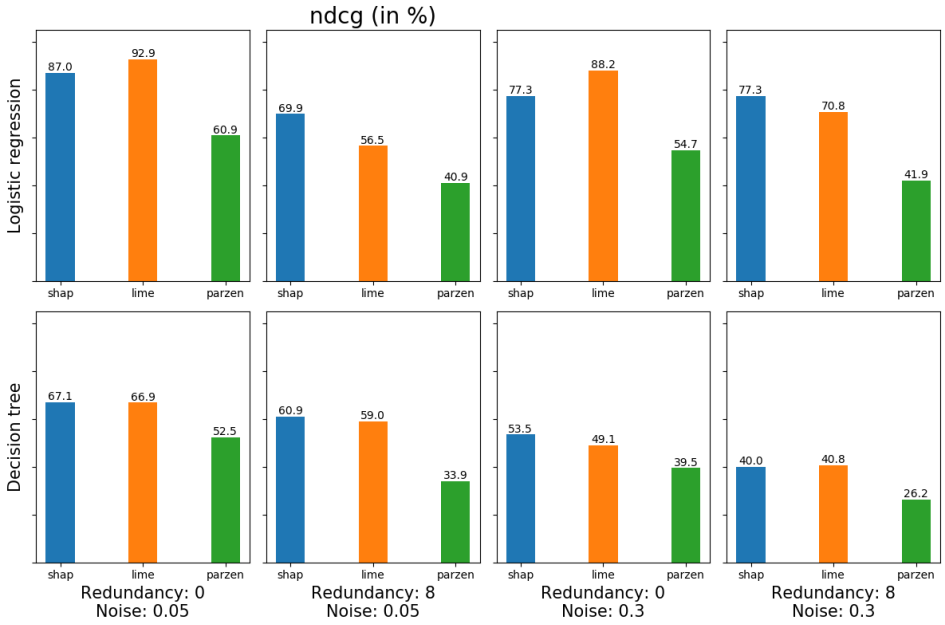


Figure 7: Experiment 5.2 ndcg metric %

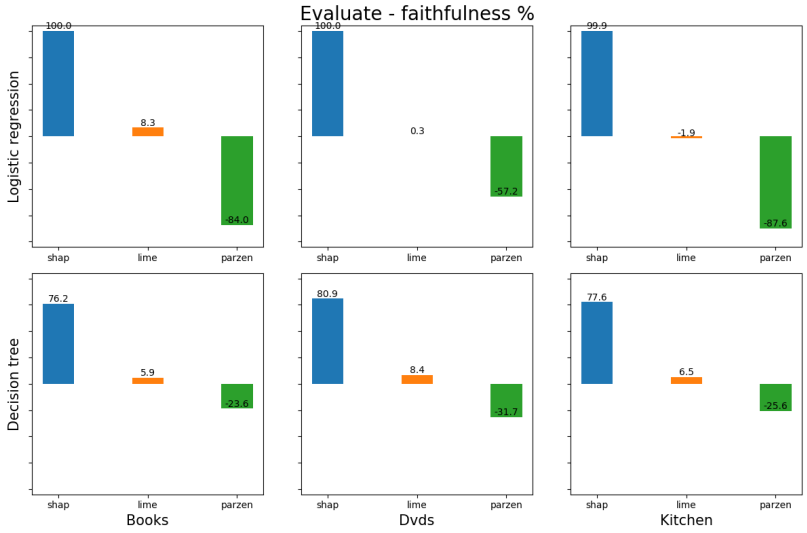


Figure 8: Experiment 5.2 faithfulness metric %

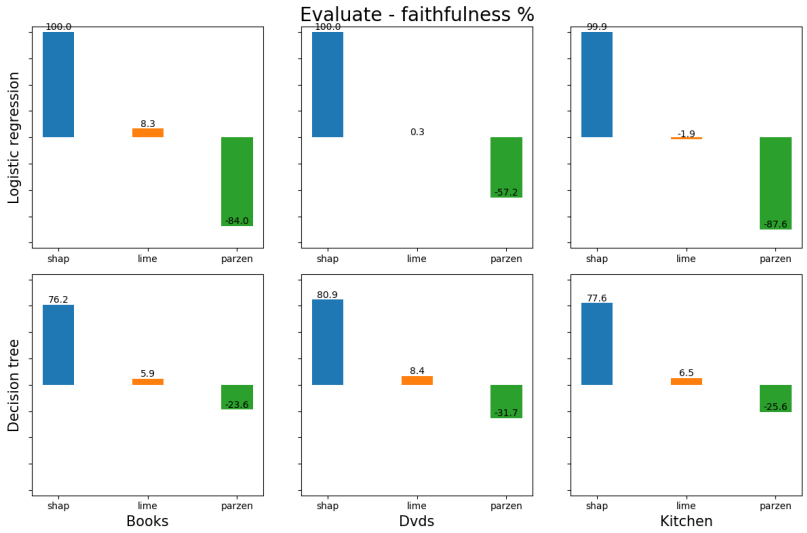


Figure 9: Experiment 5.2 faithfulness metric %

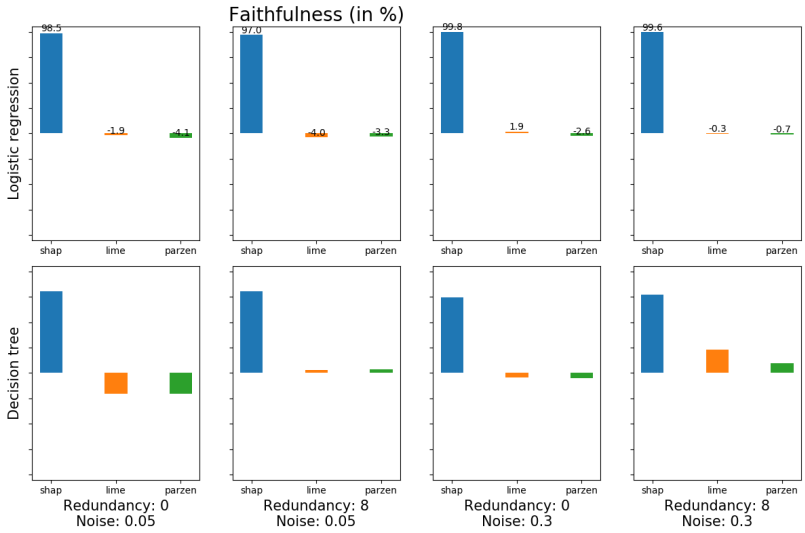


Figure 10: Experiment 5.2 faithfulness metric %

Experiment 5.3

For each instance of the data an arbitrarily selected 25% of features is deemed untrustworthy. The prediction for the model and explainer are compared before and after the untrustworthy features are discounted. The adjusted f1 score is reported in Table 3.
For all three datasets, Shap is able to produce the highest adjusted f1 score for logistic regression, support vector machine and random forest models. However, Lime achieves slightly better scores on the decision tree model. Both Lime and Shap clearly outperform the baseline Parzen with scores in the range of 35% - 66%. Interestingly, similar results appeared in experiment 5.2 where Lime delivers better results than Shap on decision tree models.
Individual scores for recall, precision and accuracy measure are included in Appendix Section 8.3. Shap produces a perfect score on accuracy. That is due to the local accuracy property, which Shapley values are guaranteed to satisfy. The accuracy scores do give a penalty to the Lime and Parzen method, as their initial classification is different from that of the predictive model.
Parzen is most heavily punished by its accuracy scores. The results show that Parzen often wrongly classifies the initial prediction. As a result, for this experiment Parzen does not compete with either Lime or Shap.

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	**96.8**	**92.3**	**98.5**	**96.7**	93.9	**96.5**	**87.0**	**98.5**	**95.9**	94.8
LIME	95.8	84.0	93.9	94.9	**97.3**	95.5	80.3	97.4	95.1	**97.7**
Parzen	53.7	61.1	59.6	65.6	54.3	53.5	47.5	47.8	55.5	48.7

	Kitchen				
	LR	NN	RF	SVM	Tree
SHAP	**97.8**	**91.4**	**99.2**	**97.6**	95.0
LIME	97.4	82.8	98.3	97.5	**97.6**
Parzen	34.8	68.3	64.5	48.0	58.5

Table 3: Experiment 5.3 - adjusted f1 score %

Experiment 5.3 Generated data

Experiment 5.3 is repeated with synthetically generated data. A random selection of untrustworthy features is discounted from the explanations. The resulting adjusted f1 scores are presented in table 4.

It is reminded that the second generated dataset has an increased amount of redundant features, the third dataset has increased amount of noise. Lastly, the fourth dataset has more redundant features and additional noise.

Where to state the generated data params?

Table 4 shows that Shap delivers the highest adjust f1 score for all datasets and models. In particular, the scores for the decision tree model are relatively higher than the Lime and Parzen explanations. Lime often produces second highest results, although exceptions exist where Parzen reaches second.

Appendix table **??** displays the individual precision, recall and accuracy scores. Lime and Parzen are able to compete with Shap in terms of precision. However, Lime and Parzen are not able to reach similar scores for recall and accuracy. Hence, Shap has dominant performance when considering the combined scores.

check ref

	Generated 1					Generated 2				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	**98.2**	**94.8**	**98.6**	**97.4**	**93.6**	**98.3**	**96.3**	**98.3**	**98.2**	**93.1**
LIME	94.5	84.8	87.8	92.4	72.9	95.6	85.9	91.4	90.5	76.8
Parzen	81.5	79.8	80.4	75.4	71.3	85.2	86.9	89.2	88.7	72.4
	Generated 3					Generated 4				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	**97.9**	**88.3**	**98.0**	**97.6**	**90.5**	**98.1**	**92.9**	**98.3**	**96.6**	**90.2**
LIME	91.5	70.1	87.0	90.5	62.8	89.0	78.5	90.7	84.3	60.6
Parzen	73.3	64.5	75.9	71.9	58.6	76.7	79.9	86.7	86.1	54.6

Table 4: Experiment 5.3 - adjusted f1 score %

6 Conclusion

The explanability of machine learning algorithms is receiving plenty attention in current literature. However, assessing and evaluating the explanation

methods is not yet an exact science. This work proposes the comparison of recently developed model explainers via simulated user experiments with real-world and synthetic data.

Consider figure 3. The original experiment 5.2 (using real-world data) boasted a perfect score for Shap on the logistic regression model. While Lime's performance is highest on the decision tree model. Improvements to the experiment have been proposed, so that the budget for the explainers is better aligned with the number of golden features from the models. Additionally, the ranking of features is taken into consideration using the ndcg metric. Results from the improved experiment are displayed in figure 4 and 5. All recall scores are lower when compared to the original experiment. The relative performance of the explainers is unchanged. The ndcg scores suggest that Lime and Parzen are penalized more once the order of features is included. Still, Lime marginally outperforms Shap on the decision tree model. Lastly, the experiment is carried out with generated data, see figure 6 and 7. Recall scores of Lime and Shap are comparable, but vary across datasets and models. The ndcg scores show even larger variation for the logistic regression model. However, for the decision tree Shap marginally outperforms Lime on the three least modified datasets.
Simulated user experiment 5.3 has been corrected using a (local) accuracy score. The resulting adjusted f1 scores on the real-world data are reported in table 3. For all models other than the decision tree Shap delivers the best results. Again, for the decision tree Lime has better performance. Admittedly, most of these scores do not differ significantly according to t-tests. Experiment 5.3 is repeated with synthetic data. Adjusted f1 scores are displayed in table 4. Shap significantly dominates the other explainers on all datasets and models.
The faithfulness metric is tested as evaluation metric for (linear) model explainers. Its scores are presented in figure 8, 9 and 10. The metric is not able to evaluate explanations of less than two features. Additionally, it shows undesirable behaviour when the metric is defined. Thus, the faithfulness metric is deemed inadmissible for the evaluation of model explainers.

check significance

In summary, improvements to previously proposed user experiments aim to better evaluate the quality of model explainers. Most often does Shap achieve the highest score for these experiments or does it come in as close second. However, exceptions do exist where Lime produces marginally better results. Notably, for the decision tree model Lime produces exceptionally good results. Parzen explanations are often the worst, but the method has

Do i answer research questions?

produced second best explanations in some cases.
According to the theory, Shap should always produce better explanations than Lime. Results in this work suggest that Shap does not yet dominate other additive explanation methods in all scenarios. Additionally, in many of the experiments has the performance of Lime been similar to that of Shap.

Mention that lime is faster?

7 Discussion & Future work

In this section several assumptions and choices made in this work will be considered. In addition, suggestions for future improvements to these experiments and future work on model explainers is put forward.
A main goal in this work is to make advances to the evaluation of model explainers. As the literature has not yet reached consensus of how to measure the explainers' quality. To this end, this research considers feature inclusion in the form of precision and recall. The proposed ndcg measure is able to include the ranking of features by their importance. However, ndcg can only be calculated if the true order of features is known. That is the case for logistic regression, the prediction for an instance is formed by the global coefficients. However, for the decision tree model only **global** feature importance is known. For experiment 5.2 the true ordering of features was considered using the global variable importance. To calculate the ndcg, that ordering is compared to the ordering of features in the explanation. While using the global feature importance is not a perfect representation, experiment 5.2 aims to show that including feature ranking is an improvement over just the recall scores.
The faithfulness metric was tested in the three versions of experiment 5.2. This research has found that in its current implementation, the metric is not admissible for the evaluation of model explainers. Since the metric uses correlation, it does not show desirable behaviour for vectors with a length of two or less features. Additionally, the metric shows deviating behaviour when explanations (or changes in model predictions) are minuscule. Nonetheless, it is the only measure that actually considers each attribution of model explainers. If these flaws can be overcome, the metric may become the most robust model explainer evaluation metric. Furthermore, should any evaluation metric for model explainers be found (especially if it can integrate attribution), it would substantially improve the field of post hoc explainers.
Both experiment 5.2 and 5.3 have been applied to generated data, so that the degree of dependency and noise can be controlled. Data generation was

rewrite

implemented using sklearn's make_classification function. However, the generated data could have been controlled even further by using an option such as 'SymPy'. Data generation using symbolic expressions is described in this post by T. Sarkar[9]. With this method, the explainers could be tested on their handling of noise and dependencies even better.
An important factor that has been omitted in this research is the speed and computational effort required for the explanation methods. While I would suggest that speed is not the most important consideration for model explainers, it could prove valuable noting that Lime's performance is comparable to Shap's in the experiments in this work. Note, that this is the case for Shap Kernel. Lime tabular was up to several times faster than Shap Kernel in the current implementation (with Shap's background data summarized in 10 clusters).
Lastly, while this research is aimed at the use of model explainers, others suggest that explanations can be harmful due to their unknown unfaithfulness. For some instances, it may simple not be (logically) possible to form an additive feature explanation. A proper explanation method should at least indicate that the unfaithfulness to the instance is too large, or even establish that an additive explanation is not possible. I propose that such an extension would greatly increase trust in model explainers.

Considerations in this section may have exposed remaining challenges for model explainers. To make things worse, I propose several improvements to and future work for model explainers.
Lime and Shap require the user to specify the number of features that should be included in the explanation. Given this length, the user should be able to interpret the explanation. Now imagine an instance where one additional feature could significantly decrease the unfaithfulness of the explanation. Then, certainly, the user would want to include that additional feature, even if the maximum number of features is exceeded. Creating a test to determine the optimal number of included features or improvement of the explanation with each additional feature, would be a promising addition to model explainers.
As mentioned in the literature review, linear explanations for highly non-linear models may not be sufficient. To this end Aas et al. [3] have proposed an extension to Shap that handles dependent features. A drawback is that explanations become harder to interpret, as they can only be considered as a

[9] https://towardsdatascience.com/random-regression-and-classification-problem-generation-with-symbolic-expression-a4e190e37b8d

cluster of dependent features. I propose that different interpretable models may be a suitable alternative to the Shap extension. For example, a (small) decision tree instead of a sparse linear model for the explanations may provide a solution to non-linearity and simplicity. In the end, one desires from an explanation that it is faithful to the model and also easy to interpret. However, increasing the length or complexity for better faithfulness compromises in interpretability. Likewise, oversimplifying an explanation will most likely result in a less faithful explanation. To summarize, in the field of model explainers a tension between interpretability and faithfulness has formed. Promising future work would progress in both these concepts.

References

[1] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier.

[2] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

[3] Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values.

[4] Camburu, O. M., Giunchiglia, E., Foerster, J., Lukasiewicz, T., & Blunsom, P. (2019). Can I Trust the Explainer? Verifying Post-hoc Explanatory Methods.

[5] Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q., Luss, R., Mojsilovic, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varschney, K., Wei, D. & Zhang, Y. (2019). One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques.

[6] Blitzer, J., Dredze, M., Pereira, F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification.

[7] Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2019). How can we fool LIME and SHAP? Adversarial Attacks on Post hoc Explanation Methods.

[8] Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent Individualized Feature Attribution for Tree Ensembles.

[9] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. R. (2010). How to explain individual classification decisions.

[10] Shapley, L. S. (1953). A value for n-person games.

[11] Štrumbelj, E., Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions.

8 Appendix

8.1 Lime

Lime finds the explanation that is the most locally faithful, while still being interpretable. For a linear explanation, local faithfulness $L(f, g, \pi_x)$ is formally defined by:

$$L(f, g, \pi_x) = \sum_{z,z' \subseteq Z} \pi_x(z) \ (f(z) - g(z'))^2$$

Where z and z' are perturbed samples.
The (heuristically chosen) kernel is defined as:

$$\pi_x(z) = exp\left(\frac{-D(x,z)^2}{\sigma^2}\right)$$

With D is some distance function with width σ. The distance function for text classification is cosine distance.

8.2 Shap

Subsection will include any method/formulas/definitions that would be too much for method section

Desirable properties

Desirable properties for additive feature attribution methods.

Property 1: Local accuracy

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

Property 2: Missingness

$$x'_i = 0 \Longrightarrow \phi_i = 0$$

Property 3: Consistency
If $f'_x(z') - f'_x(z' \backslash i) \geq f_x(z') - f_x(z' \backslash i)$ for all inputs $z' \in \{0,1\}^M$, then

$\phi_i(f', x) \geq \phi_i(f, x)$
A unique additive feature explanation model follows from these properties:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!}[f_x(z') - f_x(z' \setminus i)]$$

Shap value estimation

how Shap estimates Shapley values with higher sample efficiency [3]

8.3 Supplementary results

Experiment 5.3: real-world data

The individual scores for precision, recall and accuracy for experiment 5.3 on the real-world data are presented in the tables below.

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	95.2	88.0	98.6	95.5	92.3	94.5	81.7	98.4	94.5	93.6
LIME	95.2	86.3	96.2	95.3	96.4	94.3	82.3	97.5	94.4	96.5
Parzen	90.5	83.8	91.2	90.1	85.9	88.3	75.3	88.2	87.3	84.6

	Kitchen				
	LR	NN	RF	SVM	Tree
SHAP	96.6	89.2	99.4	96.5	94.2
LIME	96.5	88.0	98.3	96.2	96.4
Parzen	91.5	82.6	90.6	89.7	86.6

Table 5: Experiment 5.3 - real-world precision %

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	98.5	97.2	98.4	97.9	95.6	98.5	93.1	98.7	98.0	96.2
LIME	97.9	97.5	98.1	97.5	98.3	98.2	92.4	98.7	97.8	98.9
Parzen	79.4	93.0	99.2	90.2	99.6	75.6	81.3	76.7	76.1	100.0

	Kitchen				
	LR	NN	RF	SVM	Tree
SHAP	99.0	93.7	99.0	98.7	96.0
LIME	98.8	93.8	99.2	98.8	98.9
Parzen	62.8	89.1	98.2	100.0	81.5

Table 6: Experiment 5.3 - real-world recall %

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.8*	100.0
LIME	99.2	91.8	96.8	98.5	100.0	99.2	92.2	99.2	99.0	100.0
Parzen	63.5	69.2	62.7	72.8	59.0	65.8	60.8	58.2	68.2	53.2

	Kitchen				
	LR	NN	RF	SVM	Tree
SHAP	100.0	100.0	100.0	100.0	100.0
LIME	99.8	91.2	99.5	100.0	100.0
Parzen	46.8	79.8	68.5	50.7	69.8

* Shap's accuracy is slightly below 100% due to a rounding error on the 16th decimal for a single instance.

Table 7: Experiment 5.3 - real-world accuracy %

Experiment 5.3: synthetic data

The individual scores for precision, recall and accuracy for experiment 5.3 on the synthetic data are presented in the tables below.

	Generated 1					Generated 2				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	96.6	91.2	98.2	95.6	88.5	97.0	93.7	97.4	96.8	87.9
LIME	97.2	91.3	95.2	97.5	89.3	97.1	93.8	96.7	96.3	88.2
Parzen	89.8	89.0	92.1	88.0	87.4	93.5	92.4	95.0	95.0	87.2
	Generated 3					Generated 4				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	96.6	81.5	98.3	96.7	83.3	97.4	88.3	97.5	94.8	82.6
LIME	95.4	80.6	95.1	95.8	81.9	94.7	87.6	96.3	93.5	81.5
Parzen	87.4	77.9	90.9	88.4	81.4	90.7	86.6	93.8	91.8	80.8

Table 8: Experiment 5.3 - generated precision %

	Generated 1					Generated 2				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	99.9	98.8	99.0	99.3	100.0	99.7	99.1	99.3	99.6	99.8
LIME	97.1	93.7	95.2	95.1	93.7	97.4	95.5	96.3	95.2	94.9
Parzen	95.9	97.6	96.5	97.0	94.5	98.6	98.3	98.7	97.8	97.3
	Generated 3					Generated 4				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	99.3	96.5	97.6	98.5	99.4	98.7	98.0	99.2	98.6	99.7
LIME	95.3	92.4	93.1	95.1	89.4	95.6	94.7	96.7	94.9	91.9
Parzen	93.5	96.0	93.8	96.4	92.7	97.8	97.9	98.9	97.0	96.3

Table 9: Experiment 5.3 - generated recall %

	Generated 1					Generated 2				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
LIME	97.2	91.8	92.2	96.0	79.8	98.2	90.8	94.8	94.5	84.2
Parzen	88.0	85.8	85.5	81.8	78.8	88.8	91.2	92.2	92.0	79.0
	Generated 3					Generated 4				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
SHAP	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
LIME	96.0	81.5	92.5	94.8	73.5	93.5	86.2	94.0	89.5	70.2
Parzen	81.2	75.0	82.2	78.0	67.8	81.5	87.0	90.0	91.2	62.3

Table 10: Experiment 5.3 - generated accuracy %